Supplementary Material:
H2O: Two Hands Manipulating Objects for First Person Interaction Recognition

Taein Kwon[1], Bugra Tekin[2], Jan Stühmer*[3], Federica Bogo[2], and Marc Pollefeys[1,2]

[1]ETH Zürich, [2]Microsoft, [3]Samsung AI Center, Cambridge

In the supplemental material, we provide further analysis of our annotation method and evaluate different error and regularization terms. Next, we explain how the training images were prepared for object pose estimation. We then provide the implementation details, evaluation metrics and further analysis of our method for joint pose estimation and interaction recognition. We finally present further qualitative results of our method.

S.1. Analysis of the Annotation Method

Influence of different error terms. In Table S1, we analyze the influence of different error terms in our joint loss function for annotating hand & object poses. To validate the accuracy of our pose estimates, we annotate the fingertips of the hands on 500 images from 5 different views. We start with the silhouette error term, $\mathcal{L}_s$, since it optimizes the shape of the hands. We then progressively add to our loss function, the 2D joint error term ($\mathcal{L}_{2D}$), the 3D joint error term ($\mathcal{L}_{3D}$), the physical constraint error term ($\mathcal{L}_{phy}$), and the 3D mesh surface error term ($\mathcal{L}_m$). We observe that $\mathcal{L}_{2D}$ and $\mathcal{L}_{3D}$ significantly increase joint estimation accuracy. While $\mathcal{L}_m$ improves the estimates for subtle hand mesh shape and location, the improvement in joint accuracy is less pronounced. $\mathcal{L}_{phy}$ improves both the physical plausibility and the accuracy of pose annotations. Further smoothing and pose corrections give an additional boost in accuracy. Overall, all the terms of our optimization function in Eq. 1 increase the quality of our pose estimates.

$$\hat{\theta}_f = \underset{\theta}{\operatorname{argmin}} \sum_{c=1}^{N_C} (\lambda_1 \mathcal{L}_s + \lambda_2 \mathcal{L}_{2D}) + \lambda_3 \mathcal{L}_{3D} + \lambda_4 \mathcal{L}_p + \lambda_5 \mathcal{L}_{phy} + \lambda_6 \mathcal{L}_a + \lambda_7 \mathcal{L}_m \tag{1}$$

We provide below additional details for the terms of our loss function.

Silhouette error term. We use object masks obtained using a self-trained Mask RCNN [5]. For hands, we estimate hand joint 2D locations in RGB using OpenPose [1], and use them to initialize the GrabCut algorithm [10]. For each camera c, we merge the hand mask obtained via GrabCut with the object mask into a single mask, $M_{c,h,o}$, and define our silhouette error term as:

$$\mathcal{L}_s(\theta) = \sum_{i=1}^{N_V} \|M_{c,h,o}[j] - \Pi_c(H_V(\theta)[i])\| \quad \text{where } j = \underset{j}{\operatorname{argmin}} \|M_{c,h,o}[j] - \Pi_c(H_V(\theta)[i])\| \tag{2}$$

where $||\cdot||$ denotes the 2-norm, $\Pi_c(\cdot)$ gives the 2D projection of a 3D point onto the image plane, $M_{c,h,o}[j]$ returns the jth coordinate in the mask of the cth camera, and $H_V(\theta)[i]$ returns the ith vertex of the hand mesh. We compute Eq. 2 for each camera.

Physical constraint regularization. To avoid physically invalid poses (e.g. a finger inside an object), we regularize our loss function with an additional term as in [4]:

$$\mathcal{L}_{phy}(\theta_h, \theta_o) = \lambda_r \mathcal{L}_R + (1 - \lambda_r)\mathcal{L}_A \tag{3}$$

where θ_h are hand pose, θ_o are object pose parameters, $\mathcal{L}_a$ is attraction loss and, $\mathcal{L}_r$ is repulsion loss. While repulsion loss penalizes interpenetration of hand and objects, attraction loss penalizes the cases in which hand vertices are in the vicinity of the objects but the surfaces are not in contact. In our experiments, we set λ_r to 0.8.

Hand joint angle limit regularization. In our loss function, we further penalize unrealistic joint angles as in [3]:

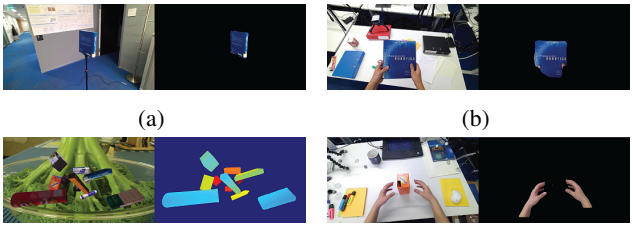
$$\mathcal{L}_a(\theta) = \sum_{k=1}^{45} (max(0, \underline{\theta_a}[k] - \theta_a[k]) + max(\theta_a[k] - \overline{\theta_a}[k], 0)) \tag{4}$$

where $\theta_a[k]$ is the k_{th} joint angle, $\underline{\theta_a}$ is the lower limit of the angle, and $\overline{\theta_a}$ is the upper limit of the angle. There exist in total 45 joint angles. As also observed in [3], the PCA space of the MANO hand model does not provide sufficient

*Work performed while at Microsoft.

Terms	$\mathcal{L}_s$	$\mathcal{L}_s + \mathcal{L}_{2D}$	$\mathcal{L}_s + \mathcal{L}_{2D} + \mathcal{L}_{3D}$	$\mathcal{L}_s + \mathcal{L}_{2D} + \mathcal{L}_{3D} + \mathcal{L}_m$	$\mathcal{L}_s + \mathcal{L}_{2D} + \mathcal{L}_{3D} + \mathcal{L}_{phy}$	$\mathcal{L}_s + \mathcal{L}_{2D} + \mathcal{L}_{3D} + \mathcal{L}_{phy} + \mathcal{L}_m$	$\mathcal{L}_s + \mathcal{L}_{2D} + \mathcal{L}_{3D} + \mathcal{L}_{phy} + \mathcal{L}_m + Smoothing$
Left Mean (std)	2.87 ($\pm$1.48)	1.25 ($\pm$1.12)	1.09 ($\pm$1.03)	1.08 ($\pm$1.02)	0.95 ($\pm$0.79)	0.95 ($\pm$0.77)	0.82 ($\pm$0.43)
Right Mean (std)	3.02 ($\pm$1.65)	1.31 ($\pm$1.03)	1.11 ($\pm$0.98)	1.11 ($\pm$1.00)	1.05 ($\pm$0.98)	1.04 ($\pm$0.94)	0.93 ($\pm$0.57)

Table S1: Impact of different error terms in our joint loss function. Errors are given in millimeters (mm). Note that all the experiments include regularization terms, $\mathcal{L}_a$ and $\mathcal{L}_p$, to avoid unrealistic hand poses.



(a) (b) (c) (d)

Figure S1: We show some examples of (a) our object images and their corresponding masks obtained by projecting the object models, (b) egocentric images and Mask R-CNN results, (c) synthetic images and the corresponding depth images for training DenseFusion, (d) hand masks obtained by GrabCut [10].

Joint	Index		Moddle		Pinky		Ring		Thumb	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
MPC(CMC)	-0.45	0	0	0	-1.5	0.5	-0.5	0.5	0	2
	-0.2	0.2	-0.2	0.2	-0.6	0.6	-0.4	0.4	-0.66	0.83
	-0	2	0	2	0	2	0	2	0	0.5
PIP(MCP)	-0.3	0.3	-0.3	0.3	-0.3	0.3	-0.3	0.3	-0.3	0.3
	0	0	0	0	0	0	0	0	-1	1
	0	2	0	2	0	2	0	2	0	1
DIP(IP)	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0
	0	1.25	0	1.25	0	1.25	0	1.25	0	1

Table S2: Hand joint limits we use in computing $\mathcal{L}_a$.

details to represent all possible hand poses. Therefore instead we use joint angle space which is more descriptive for hand poses. We calculate the limits of joint angles heuristically and give them in Table S2.

Pose prior. In order to regularize hand pose, we model the distribution of hand poses provided in the MANO dataset [9] as a multivariate Gaussian. Based on this, we define a pose prior, $\mathcal{L}_p$, which penalizes the Mahalanobis distance between both left and right hand pose θ and the learned Gaussian distributions as in [9].

$$\mathcal{L}_p(\theta) = \sqrt{(\theta - \overline{\theta})^\top S^{-1} (\theta - \overline{\theta})} \tag{5}$$

where S is the covariance of the hand pose distribution.

Pose correction. To provide an even higher quality for our pose annotations, we inspected our dataset after optimization and selected keyframes on our videos to generate smooth trajectories for hand & object poses. The number of keyframes we choose is reported in Table S3. We fix small errors of hand and object poses via interpolation based on these keyframes.

Implementation details of the annotation method. We further provide implementation details for our annotation method below.

Hand joint definition. The MANO [9] model provides hand joints for 15 locations as shown in Table S2. In order to map hand joints from the MANO skeleton to OpenPose [1] skeleton, we reorganize the order of joints and add wrist & fingertip locations by selecting corresponding points on the MANO mesh as shown in Fig. S2(b).

(a) (b)

Figure S2: (a) Object 3D models obtained with our method. We reconstruct textured 3D meshes for 8 different objects. (b) The map of the OpenPose [1] hand skeleton.

	Object	Left hand	Right hand
# keyframes	107,300	88,342	88,264
# interpolated frames	7,029	25,987	26,065

Table S3: The number of keyframes and interpolated frames for the annotations of hands and objects in our dataset.

Object pose estimation network. We use DenseFusion [14] on multi-view RGBD images to bootstrap our object pose annotations. To this end, we train on the RGB and depth images as well as the corresponding segmentation masks, given in Fig. S1(a). We train the network using ADAM [6] with a learning rate of 0.0001. We generate synthetic training images by superimposing object meshes (Fig. S2(a)) with known 6D poses on random backgrounds and cover a large variety of object poses as shown in Fig. S1(c). We create the object meshes using BADSLAM [11].

Segmentation masks. To minimize silhouette error term and train DenseFusion, we require segmentation mask for objects. To generate segmentation masks, we train Mask R-CNN [5] with a ResNet-101 backbone. We optimize the network using SGD with a learning rate of 0.001. We obtain training data for masks by projecting our 3D models onto the images as shown in Fig. S1 (a). An example result of Mask R-CNN is shown in Fig. S1 (b). We use randomly selected COCO [7] images for background augmentation (Fig. S1 (c)). As discussed in the main paper, we use GrabCut to generate segmentation masks for hands. We provide an example segmentation mask for hands in Fig. S1 (d).

Layer	Type	Filters	Size/Stride	Input	Output
0	conv	32	3 × 3 / 1	416 × 416 × 3	416 × 416 × 32
1	max		2 × 2 / 2	416 × 416 × 32	208 × 208 × 32
2	conv	64	3 × 3 / 1	208 × 208 × 32	208 × 208 × 64
3	max		2 × 2 / 2	208 × 208 × 64	104 × 104 × 64
4	conv	128	3 × 3 / 1	104 × 104 × 64	104 × 104 × 128
5	conv	64	1 × 1 / 1	104 × 104 × 128	104 × 104 × 64
6	conv	128	3 × 3 / 1	104 × 104 × 64	104 × 104 × 128
7	max		2 × 2 / 2	104 × 104 × 128	52 × 52 × 128
8	conv	256	3 × 3 / 1	52 × 52 × 128	52 × 52 × 256
9	conv	128	1 × 1 / 1	52 × 52 × 256	52 × 52 × 128
10	conv	256	3 × 3 / 1	52 × 52 × 128	52 × 52 × 256
11	max		2 × 2 / 2	52 × 52 × 256	26 × 26 × 256
12	conv	512	3 × 3 / 1	26 × 26 × 256	26 × 26 × 512
13	conv	256	1 × 1 / 1	26 × 26 × 512	26 × 26 × 256
14	conv	512	3 × 3 / 1	26 × 26 × 256	26 × 26 × 512
15	conv	256	1 × 1 / 1	26 × 26 × 512	26 × 26 × 256
16	conv	512	3 × 3 / 1	26 × 26 × 256	26 × 26 × 512
17	max		2 × 2 / 2	26 × 26 × 512	13 × 13 × 512
18	conv	1024	3 × 3 / 1	13 × 13 × 512	13 × 13 × 1024
19	conv	512	1 × 1 / 1	13 × 13 × 1024	13 × 13 × 512
20	conv	1024	3 × 3 / 1	13 × 13 × 512	13 × 13 × 1024
21	conv	512	1 × 1 / 1	13 × 13 × 1024	13 × 13 × 512
22	conv	1024	3 × 3 / 1	13 × 13 × 512	13 × 13 × 1024
23	conv	1024	3 × 3 / 1	13 × 13 × 1024	13 × 13 × 1024
24	conv	1024	3 × 3 / 1	13 × 13 × 1024	13 × 13 × 1024
25	route	16			
26	conv	64	1 × 1 / 1	26 × 26 × 512	26 × 26 × 64
27	reorg		/ 2	26 × 26 × 64	13 × 13 × 256
28	route	27 24			
29	conv	1024	3 × 3 / 1	13 × 13 × 1280	13 × 13 × 1024
30	conv	720	1 × 1 / 1	13 × 13 × 1024	13 × 13 × 10 · (3× N_c+1+N_a+N_o)
31	prediction				13 × 13 × 5 × 2 × (3×N_c+1+N_a+N_o)

Table S4: Network architecture

Figure S3: Some failure cases of our pose prediction method, due to motion blur, reflection and occlusion.

	Avg. val error (mm)	Avg. test error (mm)
Left hand joints	22.05	41.45
Right hand joints	30.12	37.21
Object vertices	36.20	47.90

Table S5: Average errors (in mm) for hand & object estimates using our pose prediction approach.

Example data. We provide in Fig. S6 and Fig. S7 additional qualitative examples for our ground-truth data, which demonstrate the high fidelity and accuracy of our dataset.

S.2. Analysis of Pose and Interaction Recognition

Implementation details for pose prediction. We provide in Table S4 the full details of our network architecture. We use YOLOv2 [8] as the backbone of our network. The input to our network model is a 416×416 image. At the output layer, we produce a 3D grid instead of a 2D grid with a dimension of $13 \times 13 \times 5$, in width, height and depth axes, respectively. We set the grid cell size in image dimensions to 32×32 pixels and in depth dimension to 15cm. We define the confidence of a prediction with a function that is inversely proportional to the distance of the prediction to the ground truth as in [13] with its default parameters. We use ADAM [6] for optimization with a learning rate of 0.0001. We randomly change the hue, saturation and exposure of our images to augment our training data.

Implementation details for interaction recognition. As shown by Eq. 9 in the main paper, the outputs from two 1×1 convolutional layers, $\mathbf{W}_\theta$ and $\mathbf{W}_\phi$, are multiplied to form a data dependent adjacency matrix, $\mathbf{S_j}$. This is followed by a softmax layer to normalize the elements in the matrix.

The dimensionality of the input to our overall TA-GCN network is $3 \times 200 \times 51$ (for $C \times T \times N$). We use 21 keypoints from left & right hand as shown in Fig. S2(b) and 9 keypoints from objects (8 corners and 1 center point of a 3D bounding box). This, in total, results in 51 keypoints fed as input to the network. For inputs larger than 200 frames, we randomly sample 200 frames. For inputs smaller than 200 frames, we pad the data by looping the clip.

Each TA-GCN block takes a $C \times T \times N$ input which is fed into 2D convolutional layer following batch normalization and a ReLu layer. Another batch normalization layer and a dropout layer are placed after the 2D convolutional layer. A skip connection is added to each TA-GCN block to learn more stable features, similarly with 2s-AGCN [12]. We set the size of the vertex neighborhood defined by the convolutional kernel as 2. The convolution for the temporal dimension is the same as ST-GCN [15].

To build our TA-GCN, we stack 10 TA-GCN blocks. The consecutive numbers of output channels for TA-GCN blocks are 64, 64, 64, 64, 128, 128, 128, 256, 256, and 256. A fully connected layer following average pooling is used as the last layer to predict the class of action labels. We train the network using SGD with a momentum of 0.9. We set the dropout rate as 0.5 and the batch size as 16. The learning rate starts from 0.005 and is divided by 10 at the 150^{th}, 200^{th} and 250^{th} epoch.

Evaluation metrics. In our paper, we use the percentage of correctly estimated poses to assess the accuracy of pose estimation. Specifically, for hand pose estimation, we use the 3D PCK metric as in [13] and consider a pose estimate to be correct when the mean distance between the predicted and ground-truth joint positions is less than a certain threshold without a rigid alignment. When using the percentage of correct poses to evaluate 6D object pose estimation ac-

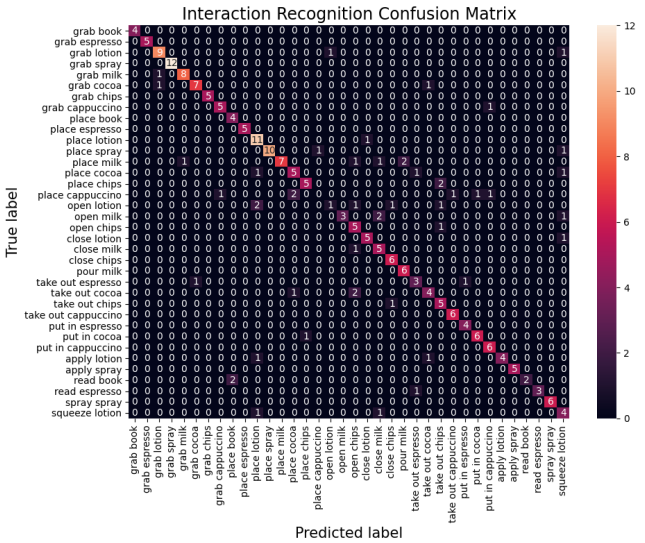


Figure S4: Confusion matrix for interaction recognition .

Figure S5: Inter-dataset examples when we train both hand poses on our dataset with our method and validate on FPHA [2]. Note that the offsets are observed due to different camera parameters across datasets. Polluted images by the magnetic sensors on the FPHA dataset detriment generalization for right hand poses.

curacy, we take a pose estimate to be correct if the 2D projection error or the average 3D distance of model vertices is less than a certain threshold (the latter being also referred to as the ADD metric).

Confusion matrix for interaction recognition. We show in Fig. S4 the confusion matrix for interaction recognition. As shown by the strong diagonal of the confusion matrix, our model is able to distinguish between different classes achieving a high accuracy.

Mean errors for hand & object keypoint prediction. We further provide average keypoint prediction errors for hands and objects in Euclidean distance in Table S5. Keypoints are selected as 21 joint locations for hands and 21 points on the bounding box (1 center point, 8 corner points, 12 midpoints of the edges) for the object. We demonstrate that, with a low error margin, our method constitutes a strong baseline for joint pose estimation of two hands interacting with objects. We provide further qualitative pose estimation results in Fig. S8.

References

[1] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *PAMI*, 43(1):172–186, 2019. 1, 2

[2] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 409–419, 2018. 4

[3] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. HOnnotate: A Method for 3D Annotation of Hand and Object Poses. In *CVPR*, 2020. 1

[4] Yana Hasson, Gul Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *CVPR*, 2019. 1

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2

[6] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 2, 3

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014. 2

[8] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. In *CVPR*, 2017. 3

[9] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied Hands: Modeling and Capturing Hands and Bodies Together. *ACM Transactions on Graphics (ToG)*, 36(6):245, 2017. 2

[10] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut" - Interactive Foreground Extraction using Iterated Graph Cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004. 1, 2

[11] Thomas Schops, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle Adjusted Direct RGB-D SLAM. In *CVPR*, 2019. 2

[12] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition. In *CVPR*, 2019. 3

[13] Bugra Tekin, Federica Bogo, and Marc Pollefeys. H+O: Unified Egocentric Recognition of 3D Hand-Object Poses and Interactions. In *CVPR*, 2019. 3

[14] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *CVPR*, 2019. 2

[15] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018. 3

Figure S6: Some examples of ground-truth data of our dataset for hand & object poses on five different camera views.

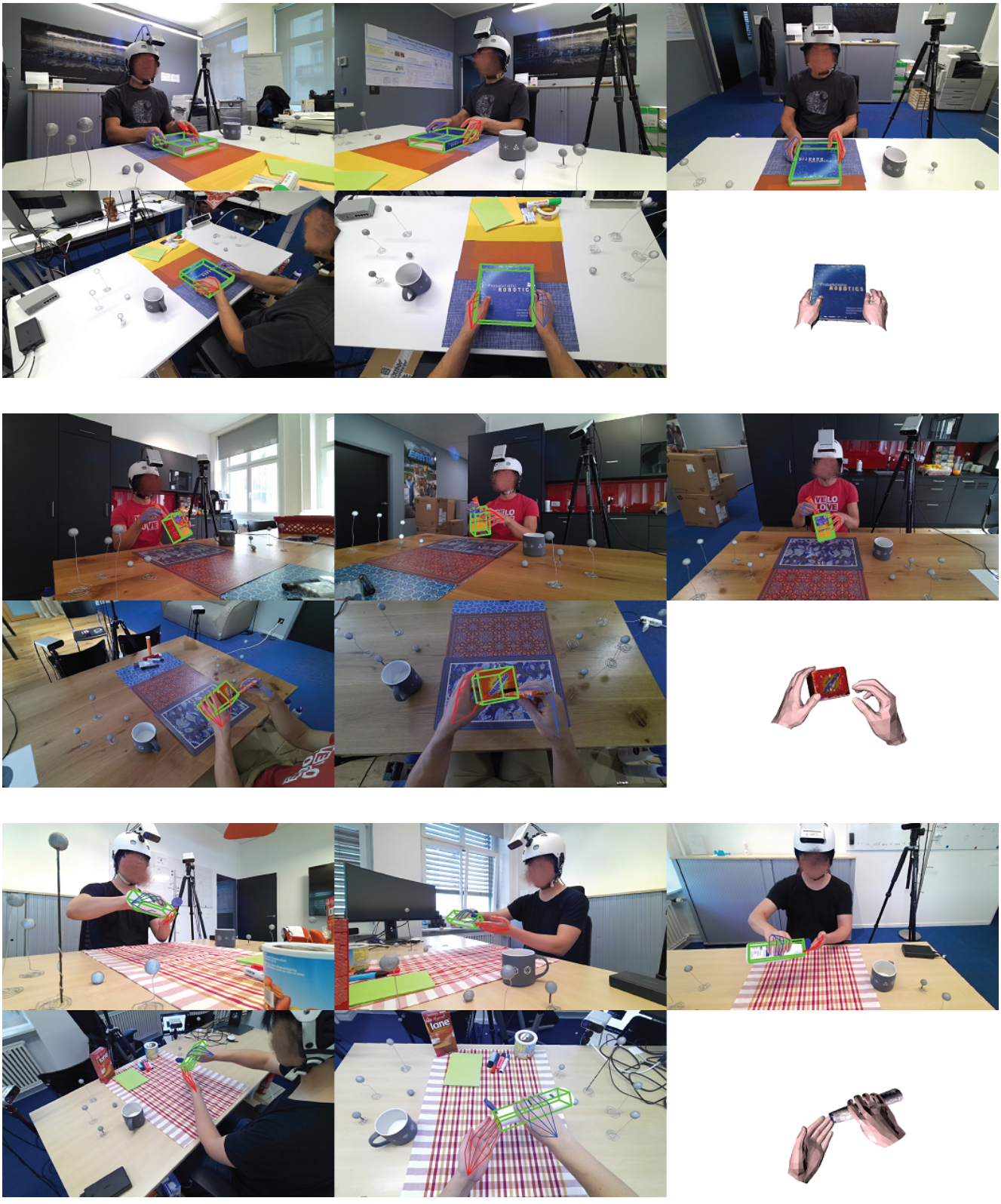


Figure S7: Some examples of ground-truth data of our dataset for hand & object poses on five different camera views.

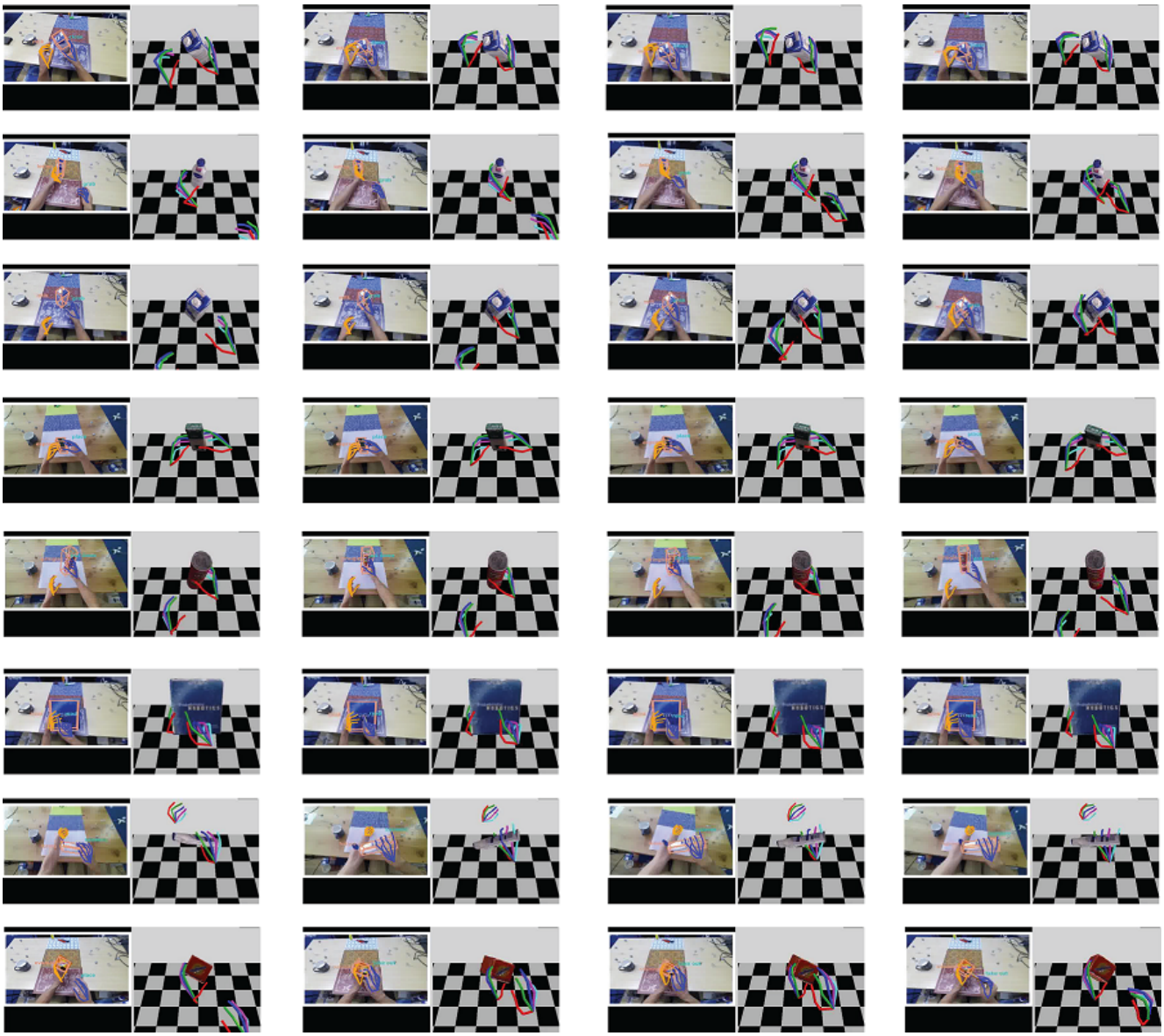


Figure S8: Qualitative results of our method that jointly estimates the poses for two hands & objects, along with action and object classes.